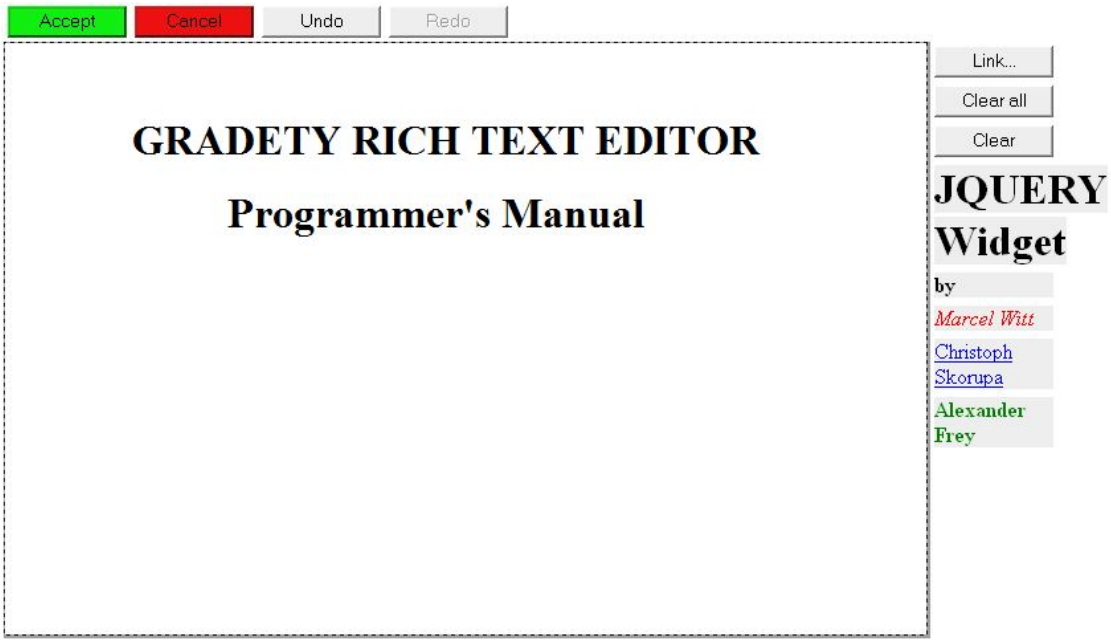
Gradety Rich Text Editor



Marcel Witt, Christoph Skorupa, Alexander Frey

Documentation for the jQuery widget created in a teamproject for FH Trier and littleweblab.com

Supervisor: Dipl. Des. Andreas Hofer

Trier, 22.02.2012

Table of Contents

1

Introduction

This documentation was created for any user of the jQuery widget described therein. The widget itself was created by three students of the University of Applied Science Trier as the result of a team project for an ambitious product named Gradety by a company called littleweblab. Goal of the project was to create a web based, client-side rich text editor which creates pure html code as you go on and type and format your text. The widget is highly customizable in its functions and its looks. How to do that and what interfaces the widget offers you to customize it is the matter of this documentation.

2

Usage

For starters the general usage and how to setup the widget is explained.

2.1 Basic Functions

The basic functions of the widget are explained by an example picture below. Note that the picture shows how the widget will look like if you don't use any customization. As you can see in the picture, somebody has already inserted some

fig. 2.1. Example of the widget in Mozilla Firefox 3.6.13 with some text and some user defined styles

text with an arbitrary formatting. On top of the text content field are buttons for accepting (and saving) changes, cancelling (and discarding) changes, undo last step and redo if you used undo. Key shortcuts for undo (CTRL-Z) and redo (CTRL-Y) are provided. On the righthand side are the buttons for adding and removing styles. The first three buttons are static and provide functions for entering a link, clearing the selection of all formatting and clearing the whole text of all formatting. Below those are buttons that are created through the input of the user when the

widget is created. By providing parameters the widget creates these buttons which can then be used to add the previewed format to selected text passages. Note that this is how the widget looks like when the option for preview is set to true. The default for this option is false. Figure 2.2 is an example how the widget looks like when the option is set to false. When you start the widget up or when you accept

fig. 2.2. Example of the widget in Mozilla Firefox 3.6.13 with style preview set to false

or cancel your editing the widget will look like in figure 2.3. By clicking on the text editing area the widget will get ready for text editing by saving the current version of your text in the undo management system and fading in the buttons.

fig. 2.3. Example of the widget in Mozilla Firefox 3.6.13 when inactive

2.2 Setup

First of all make sure that an up-to-date version of the jQuery library is provided and linked to in the file you want to use the widget in. Then you need to provide a container for the text editing. You add the widget to that container like so:

```
1 <script type="text/javascript">
2   $(document).ready(function () {
3     $('#container').gradety_rt({ formatStyles: [['h1',
4     'big red', 'head']], preview: true});
5   });
6 </script>
```

And that's already all that's needed to use the widget. In the example you can also see the options we start the widget with. For default use without customization you just need to set formatStyles to an array of tuples. Each tuple being an array with three components. The first component is the tag that will be added to the list of format buttons. Second is a list of css classes that will be added to the tag. To add more than one class just separate them with a „ “. The last component is a label for the format button. You can either provide an individual name of your choice or an empty string. If you just use an empty string the widget will create a label for the format button looking like this: „tag: classes“. For example „h1: big red“. The second option is a flag for using plain or preview buttons for the format buttons.

3

Customization

This chapter consists of a list of functions and UI elements which may be overwritten by the user. Note that the widget is set to be used with the default functions and callbacks so make sure your functions are completely compatible with the widget. Customization is not recommended for beginners or people that are new to programming with jQuery and jQuery UI. Additionally this list does not claim to provide everything you need to correctly customize the widget, make sure to take a deep look into the source code.

3.1 onCreate

Will be called when the widget startes up. Default version creates all buttons and hooks up the button and click events.

3.1.1 syntax

```
1 onCreate(event, params);
```

3.1.2 parameters

```
1 event   a javascript event object
2 params  an array consisting of the widgets
3         data object and the options
```

3.1.3 notes

By overwriting this event callback in which the ui of the widget is created you can completely change the looks of the widget. Since the default event handlers for button events are stored separately in the data object you can still use the default event handlers with your customized ui. Also remember to hook up the functions found in the widgets data.logic object with the correct ui elements.

3.2 onStartEditing

Will be called when the user clicks into the content field while the widget is not in editing mode.

3.2.1 syntax

```
onStartEditing(event, data);
```

3.2.2 parameters

```
event   a javascript event object
data    the widgets data object holding
        callbacks and ui
```

3.2.3 notes

The default event handler makes the buttons fade in and the content field change to white. Use this event handler to further customize your individual ui. Also remember to update the handler if you add any buttons or change their names.

3.3 onStopEditing

Will be called when the user clicks on the accept or cancel button in the default version of the widget.

3.3.1 syntax

```
onStopEditing(event, data);
```

3.3.2 parameters

```
event   a javascript event object
data    the widgets data object holding
        callbacks and ui
```

3.3.3 notes

Practically negates the effects of onStartEditing. If you add anything at the start of the editing process make sure to clean it up here.

3.4 onUndone

Will be called when the user clicks the undo button or uses the undo shortcut (CTRL+Z).

3.4.1 syntax

```
1 onUndone(event, data);
```

3.4.2 parameters

```
1 event   a javascript event object
2 data    the widgets data object holding
3         callbacks and ui
```

3.4.3 notes

Enables the redo button by default. Anything that comes up when the user undoes something should go in here.

3.5 onUndoDisabled

Will be called when the user reaches the final undo step. No further undoing possible.

3.5.1 syntax

```
1 onUndoDisabled(event, data);
```

3.5.2 parameters

```
1 event   a javascript event object
2 data    the widgets data object holding
3         callbacks and ui
```

3.5.3 notes

Disables the undo button by default. Anything that comes up when the user has reached the original state of his text should go here.

3.6 onRedo

Will be called when the user clicks the redo button or uses the redo shortcut (CTRL+Y).

3.6.1 syntax

```
1 onRedo(event, data);
```

3.6.2 parameters

```
1 event   a javascript event object
2 data    the widgets data object holding
3         callbacks and ui
```

3.6.3 notes

Enables the undo button by default. Anything that comes up when the user redoes something should go in here.

3.7 onRedoDisabled

Will be called when the user reaches the final undo step. No further undoing possible.

3.7.1 syntax

```
1 onRedoDisabled(event, data);
```

3.7.2 parameters

```
1 event   a javascript event object
2 data    the widgets data object holding
3         callbacks and ui
```

3.7.3 notes

Disables the redo button by default. Anything that comes up when the user has redone all his changes comes here.

3.8 onUndoStackUpdate

Will be called when the undo-redo-stack gets updated.

3.8.1 syntax

```
1 onUndoStackUpdate(event, data);
```

3.8.2 parameters

```
1 event   a javascript event object
2 data    the widgets data object holding
3         callbacks and ui
```

3.8.3 notes

When the user types something into the content field the undo-redo-stack has to be updated. New undos are possible, but no redos. Anything that comes up when the stack is updated should go here.

3.9 onDefineLink

This is the event handler which is called when adding a link to the text content. By default this happens when clicking the default link button.

3.9.1 syntax

```
1 onDefineLink(event, data);
```

3.9.2 parameters

```
1 event   a javascript event object
2 data    the widgets data object holding
3         callbacks and ui
```

3.9.3 notes

If you overwrite this event handler you should give the user a possibility to enter information for a link (e.g. URL and title). After you retrieved the information from the user you can use the method `data.logic.insertLink(url, title)` to add the link to the selected text.